CitiusTech Whitepaper

Healthcare Bigdata Tools in a Box

Authored by:

Amit Damle, Principal Architect, Big Data Practice, CitiusTech

Dattaram Mayekar, Technical Lead, Big Data Practice, CitiusTech

Omkar Kulkarni Sr. Manager, Big Data Practice, CitiusTech

Table of Content

1. Introduction 2

2. Two Minute Docker Overview 2

2.1. What is Docker? 2

2.2. What are various components of Docker? 2

2.3. Advantages of using Docker 3

3. Approach 3

3.1. H-Scale Component Image Creation 4

3.1.1. Software Prerequisite 4

3.1.2. Processing Summary of Image Creation 4

3.2. Installing Images on Customer Environment 4

3.3. Configuring and Running Images 5

3.4. Challenges and Workarounds 5

4. References 5

4.1. Docker References 5

4.2. HDP Installation 5

1. Introduction

The traditional healthcare data processing systems are inadequate to process unstructured data such as clinical notes or sensor data (high velocity data). There is a rising need for real-time analytics to help minimize patient re-admission, advance detection of abnormalities in heart beats well-in advance, detection of insurance frauds etc.

H-Scale, CitiusTech's healthcare big data platform, is a collection of loosely coupled healthcare big data processing components that consists of HL7, CCDA and claims parsers, data ingestion, real-time processing framework, data quality and reconciliation tools. With these components, organizations can address the unique requirements of the healthcare industry, such as management of structured and unstructured data, data governance capabilities (data quality, data normalization and MPI), automated and streaming analytics, real-time processing, advanced analytics, privacy and interoperability support and data security.

However, managing operational requirements of H-Scale platform such as versioning, ease of deployment, system restore, and multitenancy setup requires manual effort. Based on customer feedbacks, the following were identified as a prioritized list of requirements that need to be part of H-Scale:

- Ease of deployment
- Ease of portability to any environment e.g., Dev -> QA -> Staging -> Production
- Multitenant setup
- Adding processing nodes on demand
- Managing different component versions at the same time

H-Scale components are built using Hadoop and Big Data ecosystem components which require multi-node cluster set-up. It was difficult to fulfill the requirements without distributing H-Scale as a packaged image. An Image can be created and distributed as a virtual machine or as a container. We have considered the container option i.e. Docker due to its simplicity and light weight.

Docker container facilitates packaging of individual H-Scale components as healthcare big data processing tools which can be easily deployed on physical machines or virtual instances. Containerized components can be executed as a black box or as a platform for building quick proof of concepts.

Subsequent sections describe the process of downloading containers and installing it on physical machines on premises or on virtual cloud.

2. Two Minute Docker Overview

2.1. What is Docker?

Docker is an open source project that automates application deployment by providing capability to package an application along with runtime dependencies into a container.

2.2. What are various components of Docker?

- **Container** – Each container is based on an image that holds necessary configuration data. When you launch a container from an image, a writable layer is added on top of this image. Every time you commit a container (using the docker commit command), a new image layer is added to store your changes.

- **Image** – It is a static snapshot of the containers' configuration. An Image is a read-only layer that is never modified, and all the changes are made to the top-most writable layer which can be saved only by creating a new image. Each image depends on one or more parent images.

- **Docker file** – These are configuration files with building instructions for Docker images. Docker files provide a way to automate, reuse, and share build procedures.

2.3. Advantages of using Docker

- **Isolation** - Docker enables you to separate your applications from infrastructure, thereby enabling you to deliver software quickly.

- **Rapid application deployment** - Containers can be lifted from one environment and deployed on another environment with minimal / zero configuration changes.

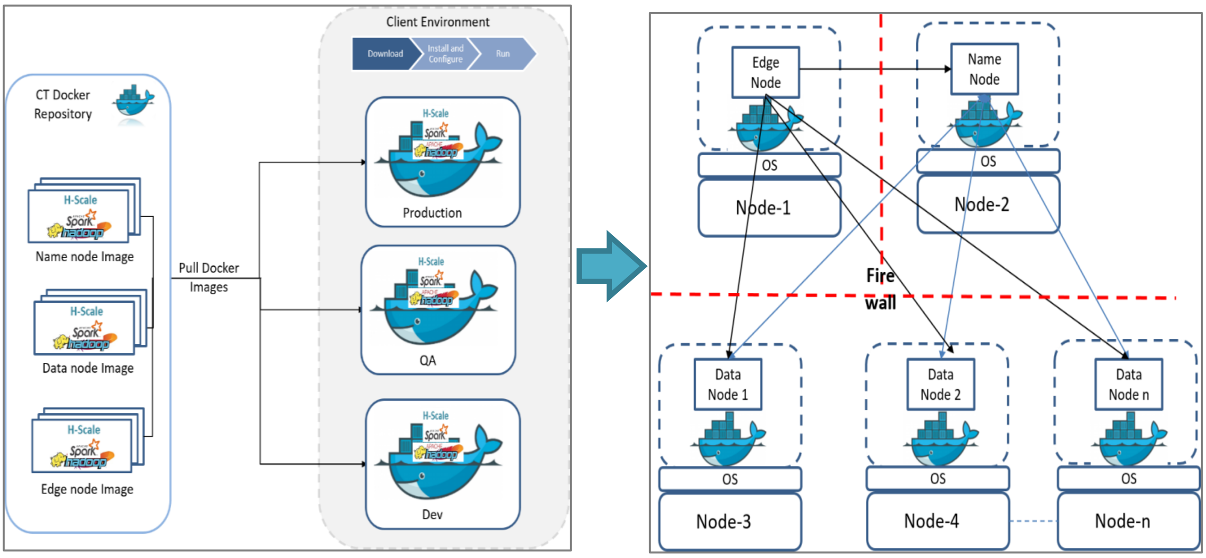
- **Portability across machines** - Application along with its dependencies can be bundled into a single container that is independent from the host Linux kernel version, platform distribution, or deployment model. This container can be transfered to another machine that runs Docker.

- **Version control and component reuse** - Tracking and managing container versions is relatively easier.

- **Improved Maintainability** - Application Dependency Management and Simplified Deployment mechanisms makes leads to ease of maintainence

3. Approach

Following diagram illustrate the process of receiving and installing H-Scale images from Docker repository hosted by CitiusTech.



Following sub-sections illustrates summary of image creation and installation on customer environment.

3.1. H-Scale Component Image Creation

The following parameters describe the bundling of H-Scale components as a Docker image and host it on CitiusTech's private Docker registry.

3.1.1. Software Prerequisite

Software specifications of default platform used for creating H-Scale image is mentioned in the following table.

#	Software	Distribution Details
1.	OS	CentOS 6.x
2.	JDK	Oracle JDK 1.8.x
3.	Hadoop	**H**ortonworks **D**ata **P**latform (HDP 2.4.x)
4.	Web server	Jetty 9.x

Note: Though default image is based on centos and HDP, depending on customer needs images can be prepared and distributed for other platform e.g., Cloudera on Centos or RHEL.

3.1.2. Processing Summary of Image Creation

1. Download the basic CentOS 6.x image from Docker registry on two different machines.
2. Spawn container with image on two machines for creating Name node and Data node.
3. Configure the pre-requisite for HDP installation like password less SSH, disable iptables etc.
4. Install and configure the ambari-server on Name node.
5. Install HDP components.
6. Install H-Scale Components e.g., Web-App, Ingestion Pipeline, Data Quality etc. with automation scripts.
7. Stop both containers and create images from them with proper naming conventions which can be used and distribute further.
8. Push image into private Docker repository hosted on CitiusTech network using the following command.

```
$ docker push https://www.citiustech.com/docker/h-scale
```

Please refer References section for detailed steps on installation.

3.2. Installing Images on Customer Environment

Customers need to be onboarded before they get access to the private Docker repository. Once registered with CitiusTech, customers can download H-Scale Docker image by providing valid credentials as shown below:

```
$ docker login https://www.citiustech.com/docker

$ docker pull https://www.citiustech.com/docker/h-scale-nn

$ docker pull https://www.citiustech.com/docker/h-scale-dn
```

3.3. Configuring and Running Images

- After installing images, it can be configured by executing H-Scale image configuration script downloaded with image.
- Administrators need to logon to physical machine on which name node container is running and provide basic configuration details e.g., hostname of the physical machines on which data node containers are installed, password to be used during SSH config etc.
- After setting the configuration, administrator needs to run H-Scale start-up script. Script will connect to all data nodes and start the containers, Hadoop and H-Scale services.

3.4. Challenges and Workarounds

Sr.No.	Challenges	Workaround
1.	SSH communication between name node and data node container happens on port 22. This port clash with the Port 22 of physical machines on which it is hosted hence two containers cannot communicate with each other.	Change the SSH port of physical machines from port 22 to any other port e.g., 2222.
2.	New container version deploys new component binary but the data present in HDFS doesn't get copied as part of the image. Hence deploying new container version will not provide access to the data loaded/processed in previous version of the container.	In current container version; HDFS folders are linked with physical file system folders thus new container versions can access the data by setting link between HDFS folders and physical folders.

4. References

4.1. Docker References

- Docker Installation
- Docker Search
- Docker PULL
- Docker EXEC
- Docker STOP
- Docker COMMIT/Image Creation

4.2. HDP Installation

- Pre-requisite
- Install Ambari
- Install HDP